Treinamento Node.js

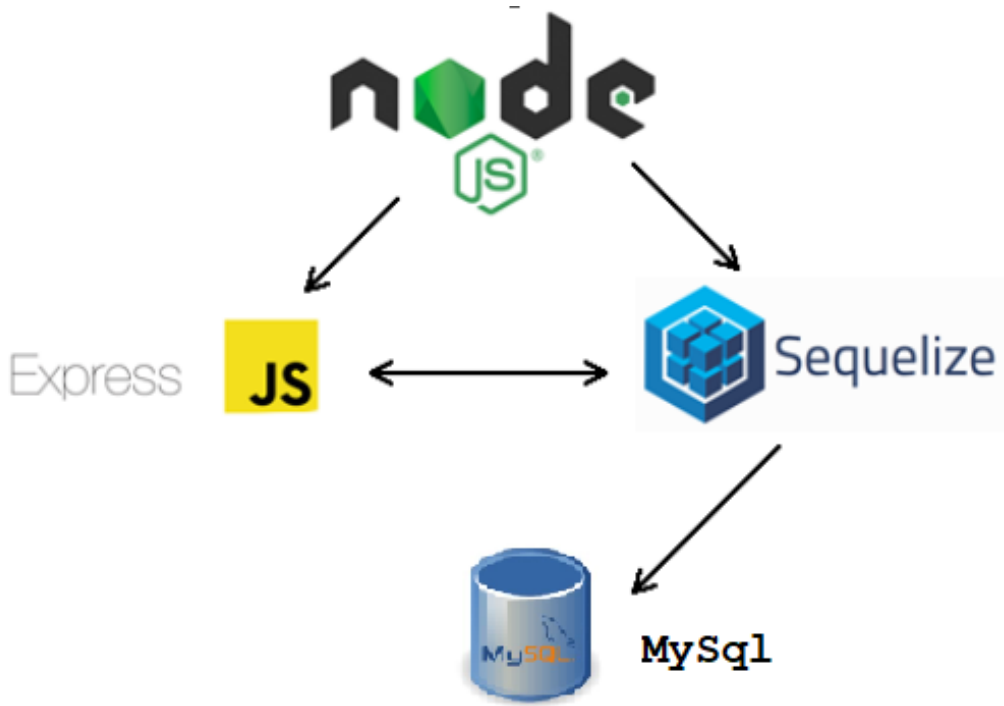
Sequelize

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

Sequelize

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

- Sequelize é um **ORM** (*Object-relational mapping*), desenvolvido exclusivamente para rodar em cima da tecnologia NODE.JS e possui hoje suporte para Microsoft SQL Server, MySQL, Postgres, MariaDB, SQLite.

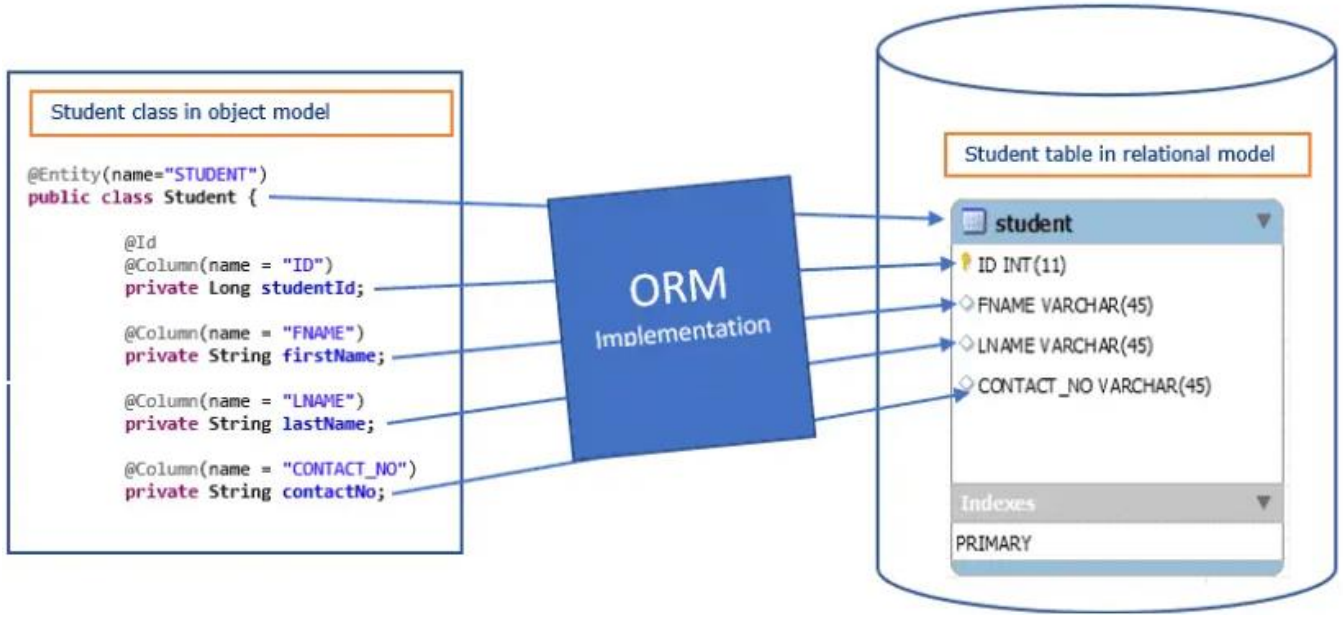


Sequelize

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

ORM

- É uma técnica usada na criação de uma "ponte" entre programas orientados à objetos e, na maioria dos casos, bancos de dados relacionais.
- O ORM é uma camada de software que conecta o seu código orientada à objetos (OOP) a bancos de dados relacionais.



Sequelize

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

Características dos ORMs

- É mais fácil atualizar, manter e reutilizar o código já que temos que escrever um modelo de dados apenas em um lugar.
  - Permite definir modelos de dados em JavaScript, o que torna a estrutura dos dados mais coesa e fácil de manter

- Não há necessidade de escrever consultas SQL.
  - Você pode interagir com o banco de dados usando JavaScript e métodos fornecidos pelo Sequelize, sem a necessidade de escrever consultas SQL manualmente

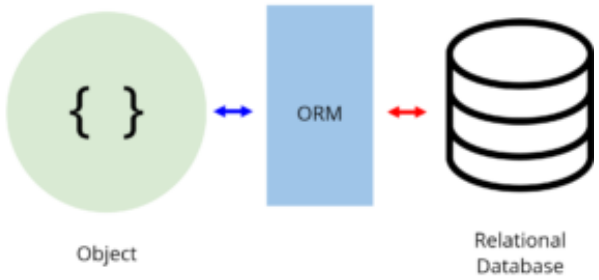
- Facilita a manutenção do esquema do banco de dados à medida que o modelo evolui.
  - Em alguns casos, pode ser necessário realizar migrações de banco de dados para acomodar alterações no modelo.

- A maior parte do trabalho é automatizada.
  - automatiza muitas tarefas de interação com o banco de dados, como criar, consultar, atualizar e excluir registros. No entanto, você ainda precisa configurar as definições do modelo e lidar com casos mais complexos, como consultas complexas que podem exigir algum conhecimento adicional.

Sequelize

2f486967682070657266f726d616c63652e2044656c6976657265642e2f486967682070657266f726d616c63652e2044656c6976657265642e2f486967682070657266f726d616c63652e2044656c6976657265642e2f486967682070657266f726d616c63652e2044656c6976657265642e2f486967682070657266f726d616c63652e2044656c6976657265642e2f48696768207

Automatizar tarefas



Sequelize

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

Instalação

- npm install -g sequelize

Documentação

- https://sequelize.org/

Sequelize

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

Utilização típica

- Conectar ao banco de dados.
- Testar a conexão.
- Criar os modelos.
- Sincronizar o modelo com o B.D.
- Executar queries.
- Validar dados.

Sequelize

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

Conectar ao Banco de Dados

```
const Sequelize = require("sequelize");

let sequelize = new Sequelize("DatabaseName", "username", "password", {
  host: "ServerName",
  dialect: "mysql",
  pool: {
    max: 5,
    min: 0,
    acquire: 30000,
    idle: 10000,
  },
});
```

Sequelize

2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070

Conectar ao Banco de Dados

```
const Sequelize = require("sequelize");

let sequelize = new Sequelize("mydb", "root", "123", {
  host: "localhost:",
  dialect: "mysql",
  pool: {
    max: 5,
    min: 0,
    acquire: 30000,
    idle: 10000,
  },
});
```

Sequelize

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

Criando um modelo (tabelas)

```
const user = sequelize.define("user", {
  id: {
    type: Sequelize.INTEGER,
    autoIncrement: true,
    primaryKey: true,
  },
  firstname: {
    type: Sequelize.STRING,
  },
  lastname: {
    type: Sequelize.STRING,
  },
  age: {
    type: Sequelize.INTEGER,
  },
  email: {
    type: Sequelize.STRING,
  },
});
```

Sequelize

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

Criando um modelo (tabelas)

```
const user = sequelize.define("user", {
  id: {
    type: Sequelize.INTEGER,
    autoIncrement: true,
    primaryKey: true,
  },
  firstname: {
    type: Sequelize.STRING,
  },
  lastname: {
    type: Sequelize.STRING,
  },
  age: {
    type: Sequelize.INTEGER,
  },
  email: {
    type: Sequelize.STRING,
  },
});
```

O Sequelize, por padrão, transforma o nome da tabela no banco de dados para minúsculo no **plural**.
Mesmo se aqui, colocássemos **User**, no banco de dados, estaria **users**.

Sequelize

2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207

Criando um modelo (tabelas)

```
const user = sequelize.define("user", {
  id: {
    type: Sequelize.INTEGER,
    autoIncrement: true,
    primaryKey: true,
  },
  firstname: {
    type: Sequelize.STRING,
  },
  lastname: {
    type: Sequelize.STRING,
  },
  age: {
    type: Sequelize.INTEGER,
  },
  email: {
    type: Sequelize.STRING,
  },
});
```

É possível:

1. Congelar o nome da tabela

```
sequelize.define("User", {
    // ... (attributes)
  },
  {
    freezeTableName: true,
  }
);
```

2. Definir manualmente o nome da tabela

```
sequelize.define("User", {
    // ... (attributes)
  },
  {
    tableName: "Employees",
  }
);
```

Sequelize

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

Sincronizando com o Banco de Dados

A sincronização é necessária para criar a tabela no banco de dados.

```
sequelize.sync()
   .then(() => { console.log("Modelo sincronizado com o banco de dados");})
   .catch((err) => {
      console.error("Erro na sincronização com o banco de dados:", err);
    });
```

Sequelize

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

Inserindo dados

```
const User = sequelize.define("user", {
  id: {
    type: Sequelize.INTEGER,
    autoIncrement: true,
    primaryKey: true,
  },
  firstname: {
    type: Sequelize.STRING,
  },
  lastname: {
    type: Sequelize.STRING,
  },
  age: {
    type: Sequelize.INTEGER,
  },
  email: {
    type: Sequelize.STRING,
  },
});
```

```
// Criando novo usuário
const jane = await User.create({
  firstName: "Jane",
  lastName: "Doe",
  age: 23,
  email: "janedoe@danedoe.test" }
);

console.log("Jane's auto-generated ID:", jane.id);
```

Sequelize

2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207

Selecionando dados

```
const User = sequelize.define("user", {
  id: {
    type: Sequelize.INTEGER,
    autoIncrement: true,
    primaryKey: true,
  },
  firstname: {
    type: Sequelize.STRING,
  },
  lastname: {
    type: Sequelize.STRING,
  },
  age: {
    type: Sequelize.INTEGER,
  },
  email: {
    type: Sequelize.STRING,
  },
});
```

```
// Buscar todos os usuários
// Equivale ao SQL:
// SELECT * FROM users
const users = await User.findAll();

console.log(users.every(user => user instanceof User)); // true

console.log("All users:", JSON.stringify(users, null, 2));


// Buscar todos os usuários, especificando os atributos
// Equivale ao SQL:
// SELECT firstname, email FROM users
const users = await User.findAll({
  attributes: ['firname', 'email']
});
```

https://sequelize.org/docs/v6/core-concepts/model-querying-basics/#the-basics

Sequelize

2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207065726f726d616e63652e2044656c6976657265642e2f48696768207

Selecionando dados

```
const { Op } = require("sequelize");
User.findAll({
  where: {
    [Op.and]: [{ a: 5 }, { b: 6 }],            // (a = 5) AND (b = 6)
    [Op.or]: [{ a: 5 }, { b: 6 }],             // (a = 5) OR (b = 6)
    someAttribute: {
      // Basics
      [Op.eq]: 3,                              // = 3
      [Op.ne]: 20,                             // != 20
      [Op.is]: null,                           // IS NULL
      [Op.not]: true,                          // IS NOT TRUE
      [Op.or]: [5, 6],                         // (someAttribute = 5) OR (someAttribute = 6)

      // Number comparisons
      [Op.gt]: 6,                              // > 6
      [Op.gte]: 6,                             // >= 6
      [Op.lt]: 10,                             // < 10
      [Op.lte]: 10,                            // <= 10
      [Op.between]: [6, 10],                   // BETWEEN 6 AND 10
      [Op.notBetween]: [11, 15],               // NOT BETWEEN 11 AND 15
    }
  }
});
```

Sequelize

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

Excluindo

```
// Exclui todos os usuários chamados "Jane"
await User.destroy({
  where: {
    firstName: "Jane"
  }
});

// Excluindo todos os registros da tabela
await User.destroy({
  truncate: true
});
```

Sequelize

Alterando

```
// Atlera todo mundo sem lastname para "Doe"
await User.update({ lastName: "Doe" }, {
  where: {
    lastName: null
  }
});
```

Atividade

- Colocar as rotas de GET e PUT da API **myAPI** funcionarem adequadamente.
- Para isso implemente, no userController.js, as funções de **getUsers** e **updateUsers**.
- Use como base a implementação do **deleteUser**.

aula5 > sequelize > myAPI > controllers > JS userController.js > ...

```
15
16  // Read (GET)
17  const getUsers = async (req, res) => {
18    // Implementar aqui
19    res.status(200).json({ mensagem: "Não implementado!" })
20  };
21
22  // Update (PUT)
23  const updateUser = async (req, res) => {
24    // Implementar aqui
25    res.status(200).json({ mensagem: "Não implementado!" })
26  };
27
```